Gradient Descent for Numerical MAX-CSP

Derek Paulsen

1 Abstract

2 Introduction

Many real world problems can be modeled as constraint satisfaction problems. The problems range from logic puzzles like sudoku to tuning search engines. Briefly, a constraint satisfaction problem is simply the task of finding a valid assignment of variables which satisfy as set of constraints. Many classic problems in theoretical computer science fit this model, such as the NP-Complete 3-SAT and circuit SAT problems. In this paper look variant of CSP, called numerical MAX-CSP. In this variant each variable is a real number and all of the constraints are linear inequalities, the goal then is to satisfy as many linear inequalities as possible.

The rest of this paper is then structured as follows. First, we give background on the problem and the basis for our solution. Next we motivate solving this problem by tying it to tuning a keyword search engine with labeled data. We then move on to describe a baseline solution by modeling the problem as mixed integer linear program and discuss the issues with this solution. Next, we present our algorithm and compare it to the baseline solution. Finally, we discuss the experimental results and give directions for future work.

3 Preliminaries

In this section we will give an overview of constraint satisfaction problems, and the specific flavor that we are attempting to tackle in this paper. We then give a brief overview gradient descent which is the basis of our proposed solution.

3.1 Constraint satisfaction problems

Constraint satisfaction problems (CSPs) encompasses many different problems. Formally a constraint satisfaction problem (CSP) is defined as a triple $\langle X, D, C \rangle$ [2], where

$$X = \{X_1, ..., X_n\} \text{ the set of variables}$$
$$D = \{D_1, ..., D_n\} \text{ the domains of each respective variable}$$
$$C = \{C_1, ..., C_m\} \text{ the set of constraints}$$

A classic example of a CSP, is the 3-SAT problem. In this case the boolean variables that appear in at least one clause would be X. The domain for each variable would be $D_i = \{true, false\}$ and the constraints would all be three variable disjunctive clauses. In this paper we focus on a subset of CSPs called Numerical MAX-CSP. In typical a CSP all constraints must be satisfied, however in MAX-CSP the goal is simply to satisfy as may constraints as possible, additionally the domains are the variables are numerical (e.g. the real numbers).

While this problem has certainly been considered before, we were only able to find one previous work that directly addresses this issue [8]. In this paper the authors propose a exact solution to numerical MAX-CSP. In particular where $D_i = \mathbb{R}$ and each constraint $C_j = a_j^T x \leq b_j, a_j \in \mathbb{R}^m, b_j \in \mathbb{R}$. That is, given a set of linear inequalities, satisfy as many as possible. In the paper the authors propose a specific algorithm for this problem based on branch and bound techniques. While the algorithm does produce optimal solutions, the algorithm is worst case exponential time. In fact, this problem is readily to formulated as a mixed integer linear program (MILP),

$$\begin{aligned} \min_{x,z} \quad & 1^T z \\ s.t. \quad & Ax - \epsilon z \leq b \\ & w \in \mathbb{R}^n \\ & z \in \{0,1\}^m \end{aligned}$$

Where ϵ is some large number. The general problem of mixed integer linear programming is NP-Hard in general (by reduction to 0-1 integer programming which is NP-Complete [6]). Hence numerical MAX-CSP is likely not to admit an efficient solution.

3.2 Gradient Descent

Gradient descent is an optimization technique which has gained much attention in recent years due to the stochastic variant being used in training neural networks for machine learning tasks such as computer vision and speech recognition. Despite this focus on machine learning, gradient descent is a general purpose optimization procedure, capable of optimizing arbitrary differentiable functions. Many variations have been proposed in recent years, however each technique uses the same basic idea. Given a function f and current point x, compute $f(x)$. Next compute the gradient $\nabla(f(x))$ and update x taking a step in a direction of $-\eta\nabla(f(x))$. Repeat this process until the minima of the function is found.

If $f(x)$ is convex, then gradient descent is guaranteed to find an optimal solution. If $f(x)$ is not convex, the gradient descent will converge to a local minimum. Although in the non-convex case, gradient descent is not guaranteed, and frequently won't, find the global minimum, it is useful for finding minima of functions which are cannot be handled by typical solvers. This makes it an invaluable tool for optimizing functions which don't admit exact optimization techniques such as those used in linear programs and quadratic programs.

4 Motivation

Numerical MAX-CSPs have a wide variety to applications in such as debugging infeasible linear programs, however we are interested in one particular problem, which is that of tuning the weights for full search engines.

Full text search engines are used in a plethora of applications. These search engines (such as Apache Lucene) are built for efficient retrieval of top-k documents based on a TF/IDF based scoring metric which is dot product between two sparse vectors $q^T d = score$[5] [3] [4]. While this default scoring gives decent results out of the box, it is frequently augmented by re-weighting the query q, with some weight vector w changing to scoring to be $(q \odot w)^T d = score$. This allows for boosting of certain terms or fields to improve the quality of search results while not changing the underlying search algorithm.

The problem we will address in this project is finding a good weight vector w. In particular our problem setting is as follows. We are giving a set of query vectors $Q = \{q_1, ...q_n\}$. For each of these query vectors q_i we are given a set of k retrieved document vectors with labels $R_i = \{(d_{i,1}, l_{i,1}), ..., (d_{i,k}, l_{i,k})\}$, $l \in \{relevant, irrelevant\}$. We wish to find a weight vector w that minimizes the number of irrelevant documents that are examined before the relevant result is found. This problem can be formulated as a mixed integer linear program (MILP) as above but with fixed $\gamma < 0$ instead of b, in particular,

$$\begin{aligned} \min_{w,z} \quad & 1^T z \\ s.t. \quad & Aw - \epsilon z \leq \gamma \\ & w \geq 0 \\ & z \in \{0,1\} \end{aligned}$$

Where each row, a, of A comes corresponds to

$$a = (d_{i,y} - d_{i,x}) \odot q_i \quad i \in \{1, ...n\} \quad x, y \in \{1, ..., k\}, l_{i,x} = relevant, l_{i,y} = irrelevant$$

That is, the pairwise difference between the components of a matching and non-matching document, multiplied by the relevant query. We note that with this setup,

$$a^T w < 0 \implies (q \odot w)^T d_{i,x} > (q \odot w)^T d_{i,y}$$

That is, the relevant document will be scored higher than the non-relevant document for the query. Solving this MILP then corresponds to minimizing the number of irrelevant documents that need to be examined in order to find the relevant document for each query.

5 Baseline Solution

As motioned above we can formulate our problem as an MILP with binary constraints. This MILP can then be feed into any supported off the shelf solver to produce an exact optimal answer. This solution is appealing because, assuming that the solver is correct, it will give an optimal solution, however it suffers from a few major drawbacks. First and foremast, depending on the input, the solver time could be exponential in the number of constraints. Moreover we found that even commercial solvers can struggle with large problem sizes, either taking large amounts of time to produce any feasible solution or having prohibitive memory requirements. For our motivating use case, we would ideally feed in many labeled queries, each producing tens of constraints, hence exponential runtime in the number of constraints greatly reduces the usefulness of the solution.

6 Our Solution

To remedy problem of runtime we propose a new solution based on gradient descent. The high level idea of the solution is simple, begin with a random weight feasible vector w and then perform gradient descent until the local minima is found. More specifically we minimize the following function,

$$f(w) = \text{HardTanh}(Aw - \gamma)^T \overrightarrow{1}$$

$$\text{Where } \gamma = -1 \text{ and for } x \in R, \text{HardTanh}(x) = \begin{cases} -1 \text{ if } x \leq -1 \\ 1 \text{ if } x \geq 1 \\ x \text{ otherwise} \end{cases}$$

Algorithm 1 OptimizeGD(f, T)

Input : The function f to optimize, a time limit T **Output :** A local minima w^*

$w^* \leftarrow \overrightarrow{0}$
while current time $< T$ **do**
  $w \leftarrow w \sim \mathbb{U}^n_{[.1,1.1]}$ // Initialize w to a uniform random vector
  for $i = 1, ...20$ **do**
    for $j = 1, ...25$ **do**
      $w \leftarrow w - \text{ADAM}(\nabla f(w))$
    end for
    if $f(w) < f(w^*)$ **then**
      $w^* \leftarrow w$
    end if
  end for
end while
return w^*

We use the ADAM optimizer [7] because we emprically found it to perform well, while giving minimal overhead.

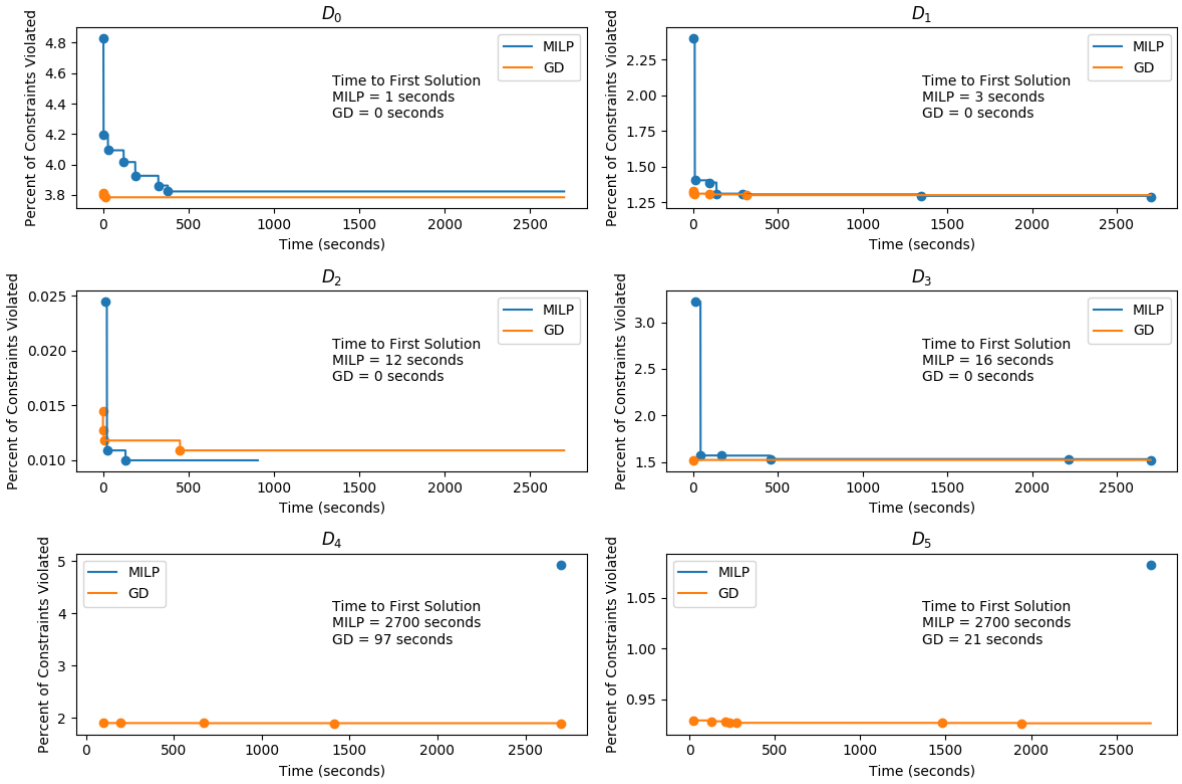
7 Experiments

We ran our experiments on a server with an AMD Ryzen Threadripper 2950X (16c/32t) processor with 64GB of RAM. Our MAX-CSP instances are generated from real world datasets, with the number of constraints ranging from 7.7K to 6.0M. For the baseline solutions we leverage Gurobi, a commercial solver with an academic license. We choose gurobi because it is consistently one of the top performing available solvers [1]. Both solutions where given a timeout of 2700 seconds (corresponding to 24 hours of CPU time).

Table 1: Dataset statistics

Dataset	# of Constraints	# of Columns	% Non-Zero
D_0	7.7K	14	66.24
D_1	19.3K	18	61.46
D_2	111.9K	18	72.83
D_3	196.2K	30	51.62
D_4	3.7M	124	25.09
D_5	6.0M	22	68.58

Figure 1: Comparison of MILP vs. Gradient Descent (GD)



8 Discussion

In this section we will discuss the experimental results. There are two major points. We first discuss the quality of solutions produced. Next, we discuss scalabilty of solutions.

8.1 Quality of Solutions

Much to our surprise, the quality of solutions produced by gradient descent (GD) was very similar if not better than the MILP solution. We attribute this to the fact that only one dataset was solved to optimality (D_2), in fact, we let D_0 run for over 12 hours and MILP still hadn't found an optimal solution. This was rather suprising because, D_0 has less an order of magnitude fewer constraints than D_2. This underlines up a key trade off between the solutions, namely, quality of solutions vs. runtime predictability. While the MILP solution will produce optimal solutions given enough time, it is very hard to predict how long it will take to produce an optimal solution. On the otherhand, there are no guarantees that can be made about GD in terms of the quality of solutions but its runtime is very predictable, scaling roughly linearly with the number of constraints times the number of columns.

8.2 Scalability

This then brings us to our second point which is the scalability of the solutions. To begin, from D_4 and D_5, we can clearly see that GD has superior scaling properties when compared to MILP in terms of runtime. MILP timed out on both datasets before even begining to refine the solutions produced. We attribute this again to the vastly different complexity of the two solutions. While GD has linear complexity in the number of constraints, MILP has best case polynomial complexity in the number of constraints. For small problems this not an issue, but as the problem size increases MILP quickly becomes impractical for even approximating the optimal solution.

9 Future Work

We identify multiple directions for future work, which fall into two categories, optimization techniques, and problem applications.

9.1 Optimization Techniques

In this paper we have only taken a small look at the possible optimization techniques that could be applied. The most obvious variation that could tried is to do stochastic gradient descent. This variation could have two major potential benefits. First, it can decrease both runtime and memory requirements of the solution further by no longer requiring the entire dataset be processed on each gradient update. Second, the stochastic version may be able to find better solutions as it is much more likely that it could escape worse local minima, instead of relying mostly on having a good starting point.

9.2 Problem Applications

We restricted our problem setting to numerical MAX-CSPs with linear constraints, however in principle there is no reason why this technique could not be applied to arbitrary constraints. In particular, non-convex constraints would potentially be solved by applying similar techniques, which current solvers are not able to handle in an capacity. We believe that it is likely that modifying the optimzation procedure, such as using stochastic gradient descent, will likely be required to provide good approximations of such functions.

10 Conclusion

In this work we have taken brief look at a particular kind of CSP, namely, numerical MAX-CSP's. We motivated this problem with a real world use case for tuning full text search engines. We then proposed a solution based on gradient descent and gave preliminary evidence to show that it has some key advantages over the baseline MILP based solution of the problem, especially for large problem sizes where commercial solvers are too slow to practical. We believe that our solution shows promise for finding approximate solutions to problems which were not previously tractable with traditional MILP based solutions. Future work should continue to explore the possible variations and possibly apply similar techniques to related problems.

References

[1] Decision tree for optimization software. URL http://plato.asu.edu/bench.html.

[2] Constraint satisfaction problem, Feb 2022. URL https://en.wikipedia.org/wiki/Constraint_satisfaction_problem.

[3] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. *Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03*, 2003. doi: 10.1145/956863.956944.

[4] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, 2011. doi: 10.1145/2009916.2010048.

[5] A. Grand, R. Muir, J. Ferenczi, and J. Lin. From maxscore to block-max wand: The story of how lucene significantly improved query evaluation performance. *Lecture Notes in Computer Science*, page 2027, 2020. doi: 10.1007/978-3-030-45442-5_3.

[6] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, page 85103, 1972. doi: 10.1007/978-1-4684-2001-2_9.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

[8] J.-M. Normand, A. Goldsztejn, M. Christie, and F. Benhamou. A branch and bound algorithm for numerical max-csp. *Constraints*, 15(2):213237, 2009. doi: 10.1007/s10601-009-9084-1.